可視化ライブラリを触ってみよう

~~Highcharts.js~~

NVD3.js

2016年7月16日

先端IT活用推進コンソーシアム
クラウド・テクノロジー活用部会

予定していたHighcharts.js とは

Highcharts is a product that was created by the Norway-based company, Highsoft. Highcharts was released in 2009, and it is a charting library written in pure JavaScript.
The product is developed in Vik, Norway and has been regularly featured in the national media, such as Finansavisen and Dagsrevyen.

Wikipediaより

- **特徴**
  - **最強にキレイなグラフが書けるJavaScriptライブラリ**
  - **折れ線グラフ、棒グラフ、円グラフ、e.t.c…**
    - http://www.highcharts.com/demo
- **注意：商用利用は有料 → 社内で気軽に使えないため変更**
  - **企業のWebサイト、イントラネットでの使用は有料**
  - **１Webサイトライセンス：約２万円**
  - **１開発者ライセンス：約５万円**

ナゼ、わざわざWebでやるのか？

- ExcelでもRでも、グラフを作ることができるけど
  - しかも、実はExcelやRの方が簡単
- Webでやるメリット
  - そのままWebで公開できる
    - しかも、リアルタイムにデータを反映できる
  - インタラクティブなグラフが作れる
    - 他のWebページと連携できる
  - センス次第で、いくらでも拡張できる

タイムスケジュール

- 14:00〜14:30
  - NVD3.js＋D3.js概要
- 14:30〜16:00
  - NVD3.jsを触ってみる
- 16:00〜17:45
  - グループワーク
    - センサからのデータを可視化してみよう
- 17:45〜18:00
  - 可視化したものを見てみよう

本日の目標

- NVD3.jsの使い方を理解する
  - できるだけ微調整で動作するサンプルを提供
  - もっと色々やりたい人はJavaScriptを勉強してください
- センサからのデータを可視化してみる
  - リアルタイムにグラフを作成

- 注意事項
  - 今日は、WindowsもMacも、**Firefox**を使うこと

- 前回（D3.js）の資料
  - http://aramoto.sakura.ne.jp/20160618_D3js/

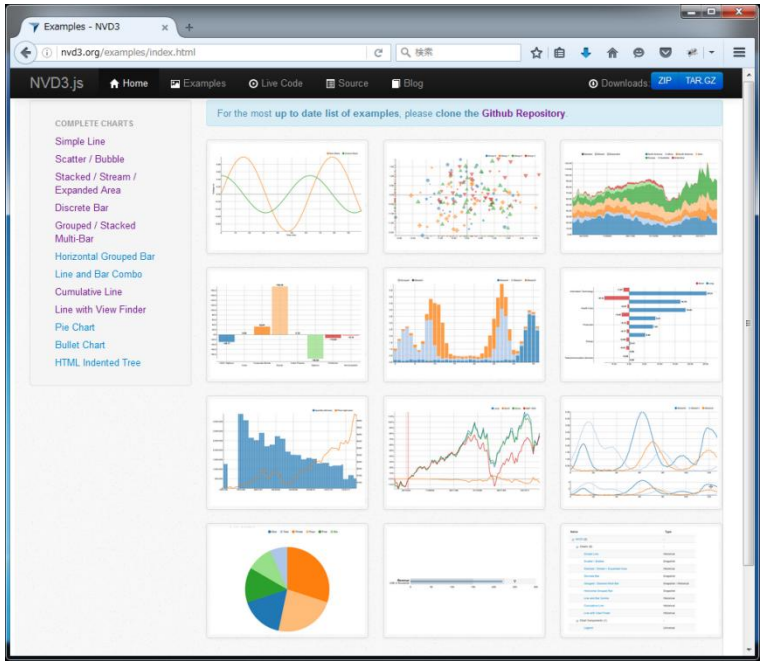
NVD3.js入門

NVD3.js とは

- **特徴**
  - **D3.jsをベースとしたグラフ描画ライブラリ**
  - **折れ線グラフ、棒グラフ、円グラフ、e.t.c…**

  http://nvd3.org/examples/index.html
- ドキュメント

  http://nvd3-community.github.io/nvd3/examples/documentation.html



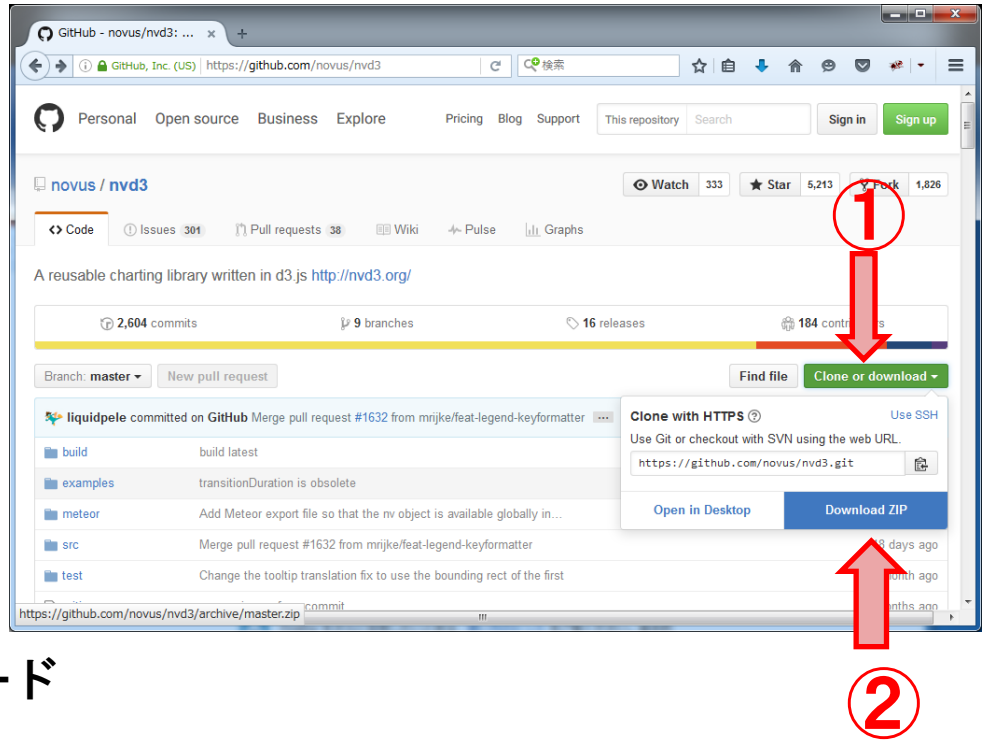
- Apache2.0ライセンス

要求するのは、ユーザーがそのソフトウェアにApache Licenseのコードが使われていることを知らせる文言を入れることだけである。

Wikipediaより

開発環境の準備－１

- NVD3.js をダウンロード
  - https://github.com/novus/nvd3
    - 「Clone or download」→「Download ZIP」
    - build/nv.d3.min.css と build/nv.d3.min.js を取り出す
    - 今回は、すでにnvd3ディレクトリに入っているので不要



- D3.js をダウンロード
  - http://d3js.org/
    - ver3のD3.jsをダウンロード
      - http://d3js.org/d3.v3.js
    - 最新のver4.1.0では、NVD3.jsが正常に動作しなかった
    - 今回は、すでにd3ディレクトリに入っているので不要

開発環境の準備－２

- テキストエディタ
  - いつも使っているテキストエディタでOK

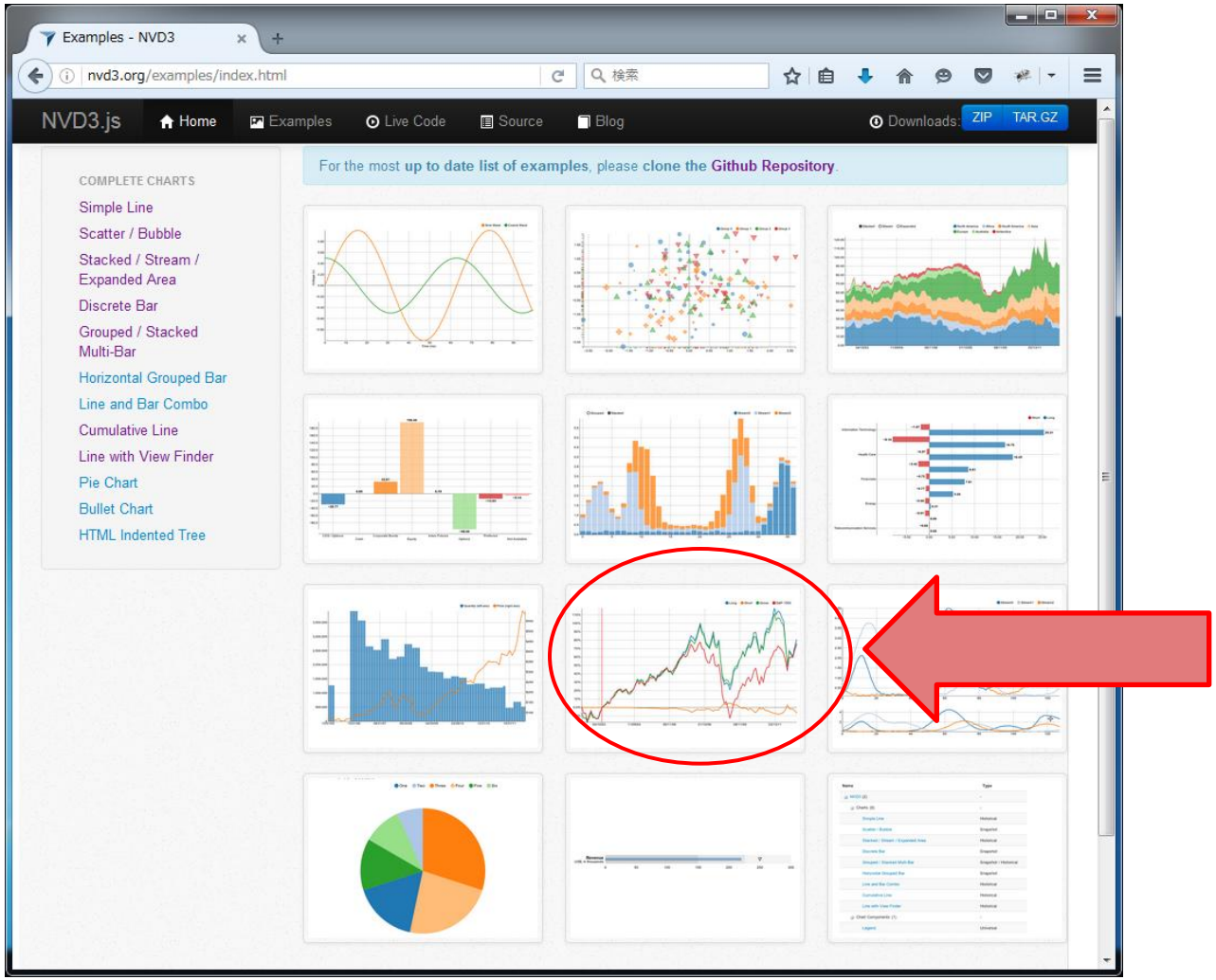
- 注意：Macのテキストエディットは要設定変更
  - Mavericks以降、ダブルクォートが自動的に変換される
  - この機能を無効にしておくこと

開発環境の準備－３

- ブラウザ
  - [F-12]を押せば、開発ツールが起動する
  - **Firefox ← 今日はこれを使用してください**
    - オススメアドオン：Firebug
      - 「ツール」→「アドオン」→「Firebuf」で検索
      - 「ツール」→「Web開発」→「Firebug」→「Firebugを開く」
  - **Safari**
    - 「環境設定」→「詳細」→「ﾒﾆｭｰﾊﾞｰに”開発”ﾒﾆｭｰを表示」
    - ローカルファイルにアクセスできない場合
      - ﾒﾆｭｰ「開発」→「ローカルファイルの制限を無効にする」をﾁｪｯｸ
  - IE（9以降）
    - ローカルだとうまく動作しない
      - データにアクセスしている部分をjQueryにすれば、動くらしい
      - PCにApacheを導入した方が早い
  - Chrome：「ツール」→「デベロッパーツール」
    - ローカルのファイルを参照するため、起動オプションを追加
      「--allow-file-access-from-files」

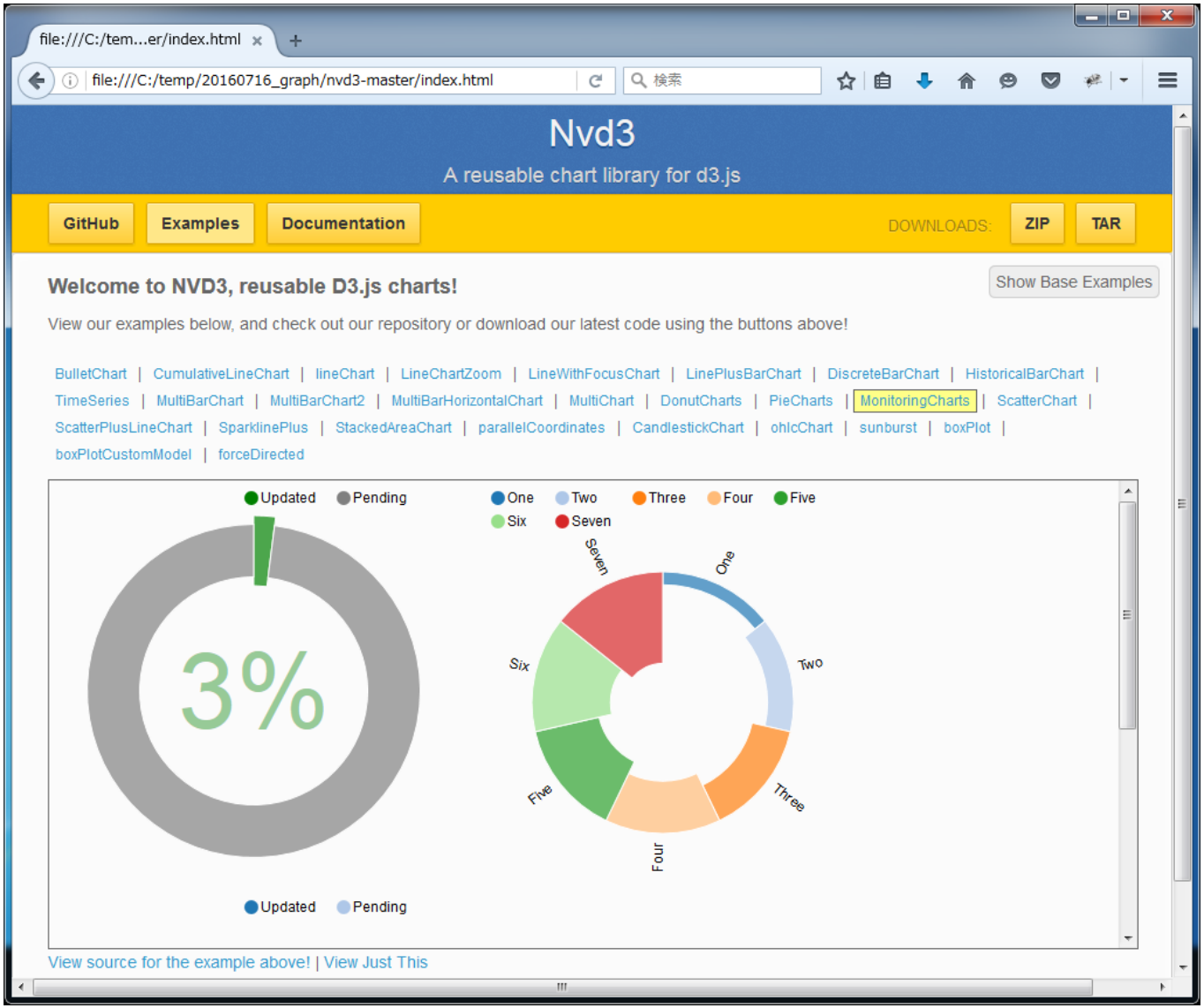
今日のメインテーマ

- センサーデータで折れ線グラフを描く
  - http://nvd3.org/examples/index.html



その他のサンプル

- ダウンロードしたnvd3-master/index.htmlを開く



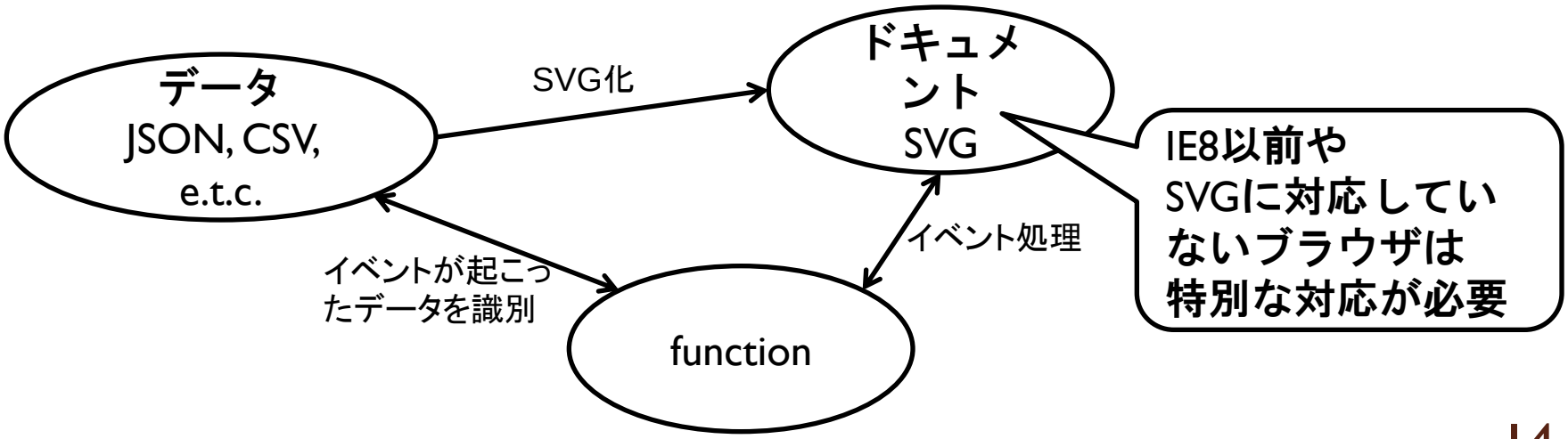
まずは、
前回やったD3.jsの復習

D3.js とは

D3.js（またはD3:Data-Driven Documents、旧:Protovis[1]）は、2011年に開発が始まった[2]ウェブブラウザで動的コンテンツを描画するJavaScriptライブラリである。World Wide Web Consortium準拠のデータ可視化ツールとして、Scalable Vector Graphics（SVG）、JavaScript、HTML5、Cascading Style Sheetsを最大限に活用している。その他多くのライブラリとは対照的に、最終的に出力された結果に視覚的な調整ができる。[3]

ウィキペデアより

- **データをドキュメント化して、関係も保持**
  - **SVGを効率良く生成するためのライブラリです**



SVG(Scalable Vector Graphics)について

- Wikipedia
  - http://ja.wikipedia.org/wiki/Scalable_Vector_Graphics

> **Scalable Vector Graphics**（スケーラブル・ベクター・グラフィックス、**SVG**）は、XMLをベースとした、2次元ベクターイメージ用の画像形式の1つである。アニメーションやユーザインタラクションもサポートしている。SVGの仕様はW3Cによって開発され、オープン標準として勧告されている。

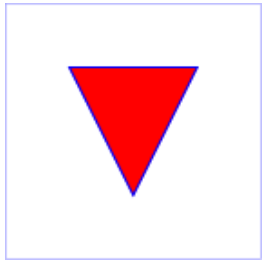
- SVG仕様
  - https://triple-underscore.github.io/SVG11/

- 使用上の注意
  - 順番通りに上に重ねて描画
  - 対応していないブラウザだと、何も表示されない
  - 注意：ブラウザによっては、微妙に見え方が違うかも

SVG(Scalable Vector Graphics)について

- **代表的な図形：sample_svg.html参照**

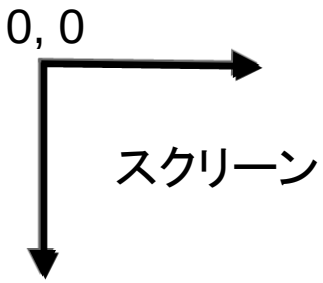
```
<rect  x="200"  y="50"  width="400"  height="200"
    fill="yellow"  stroke="navy"  stroke-width="10"  />

<circle  cx="600"  cy="200"  r="100"  fill="red"  stroke="blue"
  stroke-width="10"  />

<path  d="M 100 100 L 300 100 L 200 300 z"  fill="red"
    stroke="blue"  stroke-width="3" />

<line  x1="100"  y1="300"  x2="300"  y2="100" stroke="green"
  stroke-width="5" />

<text  x="250"  y="150"  font-family="Verdana"  font-size="55"  fill="blue"
  >  Hello, out there  </text>
```



  - **注意：後ろに書いたものが上にくる**
- **D3.jsは、データからSVG（HTML）を自動生成する**

SVGで絵を描く上での注意

- 座標は、左上がx=0, y=0
  - 右下に向かって、数値が増えていく



- 色は、名前か”#000000” で指定
  - http://www.hi-ho.ne.jp/douton/htmlcolor.html
    - 赤：red　　#ff0000
    - 緑：green　　#008000
    - 青：blue　　#0000ff
    - 黒：black　　#000000
    - 白：white　　#ffffff
    - 灰：gray　　#808080

step1.html：SVGサンプル

```
<!DOCTYPE html>                    ← HTML5で記述することを宣言
<html lang="ja">                   ← html部
<head>                             ← ヘッダ部
<meta charset="SJIS">   ← このファイルの文字コードはSJIS
<title>Sample SVG</title>
</head>
<body>                             ← ボディ部
<svg width="1000" height="1000">          ← SVG領域を1000x1000で確保
  <rect   x="200"  y="50"   width="400" height="200" fill="yellow"  stroke="navy" stroke-width="10" />
                          ← SVG 領域内に矩形を描く
  <circle  cx="600"  cy="200"  r="100"    fill="red"  stroke="blue"
      stroke-width="10" />                          ← SVG領域内に円を描く
  <line   x1="100" y1="300" x2="300"   y2="100"    stroke="green" stroke-width="5" />
  <path   d="M 100 100 L 300 100 L 200 300 z" fill="red" stroke="blue" stroke-width="3" />
  <text   x="250"  y="150"  font-family="Verdana"  font-size="55"  fill="blue" >
Hello, SVG World</text>
</svg>
</body>
</html>
```

課題－１

- step1.html を修正
  - 文字を箱の中に収まるようにする
  - 文字を自分の名前に修正
    - 保存する文字コードに注意
  - 車の絵にする

課題－１、回答例

- http://free-illustrations.gatag.net/2015/03/08/160000.html

←SVG形式で
ダウンロード可能

D3.jsの超概要 http://ja.d3js.node.ws/

- セレクタ（W3C Selectorsを参照）
  - d3.select(“#hoge”) → <xxx id=“hoge”> を対象
  - d3.select(“.hoge”) → <xxx class=“hoge”>を対象
  - d3.select(“hoge”) → <hoge>を対象

```
signal = [
    { “cx”: 100, “cy”: 100, “color”: “#0000ff”, “title”:”青”,},
    { “cx”: 200, “cy”: 100, “color”: “#ffff00”, “title”:”黄”,},
    { “cx”: 300, “cy”: 100, “color”: “#ff0000”, “title”:”赤”,},
];
```

- セレクション
  - selectAll(), enter(), exit()
  - 繰り返し処理が楽に書ける
    - d3.select("#TEXT1").**selectAll("p")**.style("color", “#000000");
- 動的プロパティ
    - svg.selectAll(".node").data(signal).text(**function(d) { return d.title; }**);
    - svg.selectAll(".node").data(signal).text(**function(d, i) { return i; }**);
- データの結合
  - 追加：svg.selectAll(".node").data(signal).**enter()**.append(“text").text(“piyopiyo”);
  - 更新：svg.selectAll(".node").data(signal).text(“hogehoge”);
  - 削除：svg.selectAll(".node").data(signal).**exit()**.remove();

NVD3.jsを使ってみる

step2.html：NVD3.js入門

```
<!DOCTYPE html>
<html lang="ja">
<head>
<meta charset="SJIS">
<link type="text/css" rel="stylesheet" href="./nvd3/nv.d3.min.css" charset="UTF-8">
<script src="./d3/d3.v3.js" charset="UTF-8"></script>
<script src="./nvd3/nv.d3.min.js" charset="UTF-8"></script>

<script src="./step2.js"></script>
</head>
<body>
<div id="view"><svg width="800" height="400"></div>
</body>
</html>
```

以降は、jsファイルの指定が違うだけ

step2.js：NVD3.js入門、前半

```
window.onload = function() {
    // グラフ化する元データ
    data1 = [{x:1, y:1}, {x:2, y:2}, {x:3, y:3}, {x:4, y:4}, {x:5, y:5},];
    data2 = [{x:1, y:20}, {x:2, y:19}, {x:3, y:18}, {x:4, y:17}, {x:5, y:1},];

    // 描画データを１つにまとめる
    var myData = [
        {       key: "data1", values: data1, color: "#ff0000",},
        {       key: "data2", values: data2, color: "#0000ff",},
    ];
    console.log(myData);            // 描画データをデバッグ出力
```

console.log を見る方法を説明

step2.js：NVD3.js入門、後半

```
    nv.addGraph(function() {
        // グラフオブジェクト生成
        var chart = nv.models
                .lineChart()  // ← multiBarChart() に変えてみる
                .useInteractiveGuideline(true)
                .showLegend(true)
                .showYAxis(true)
                .showXAxis(true);

        // グラフをHTMLの中に入れる
        d3.select("#view svg")
                .datum(myData)
                .call(chart);

        // リサイズ時の再描画
        nv.utils.windowResize(function() { chart.update()});

        return chart;
    });
}
```

課題－２

- step2.js を修正
  - .lineChart()　→ .multiBarChart() に変更
    - その他のグラフ
      - http://nvd3-community.github.io/nvd3/examples/documentation.html
      - boxPlotChart(), bulletChart(), candlestickBarChart(), cumulativeLineChart(), discreteBarChart(), historicalBarChart(), linePlusBarChart(), lineWithFocusChart(), multiBarChart(), multiBarHorizontalChart(), multiChart(), ohlcBarChart(), parallelCoordinatesChart(), pieChart(), scatterChart(), sparklinePlus(), stackedAreaChart(), sunburstChart(), axis(), boxPlot(), bullet(), candlestickBar(), discreteBar(), historicalBar(), legend(), line(), multiBar(), multiBarHorizontal(), ohlcBar(), parallelCoordinates(), pie(), scatter(), sparkline(), sunburst(), tooltip(),
  - データの値を変えて、グラフの変化を確認
  - データ件数を変えて、グラフの変化を確認

step3.js：年月日データ、前半

```
window.onload = function() {
    // データのX軸を年月日にする
    data1 = [{x:"2016/01/01",y:1}, {x:"2016/01/02",y:2}, {x:"2016/01/03",y:3}, {x:"2016/01/04",y:4},
    {x:"2016/01/05",y:5},];
    data2 = [{x:"2016/01/01",y:20}, {x:"2016/01/02",y:19}, {x:"2016/01/03",y:18}, {x:"2016/01/04",y:17},
    {x:"2016/01/05",y:1},];

    // 描画データを１つにまとめる
    var myData = [
        {       key: "data1", values: data1, color: "#ff0000",},
        {       key: "data2", values: data2, color: "#0000ff",},
    ];
    console.log(myData);            // 描画データをデバッグ出力
```

step3.js：年月日データ、後半

```
nv.addGraph(function() {
        // グラフオブジェクト生成
        var chart = nv.models
                .lineChart()
                .x(function(d) { return Date.parse(d.x) })          // X軸データの取り出し方
                .y(function(d) { return d.y })                      //Y軸データの取り出し方
                .useInteractiveGuideline(true).showLegend(true).showYAxis(true).showXAxis(true);

        // x軸の表示形式設定（年月日表記）
        chart.xAxis.axisLabel('年月日').tickFormat(function(d) {
                return d3.time.format('%Y/%m/%d')(new Date(d))
        });
        // y軸の表示形式設定（数値表記）
        chart.yAxis.axisLabel('値').tickFormat(d3.format('d'));

        // グラフをHTMLの中に入れる
        d3.select("#view svg").datum(myData).call(chart);
        // リサイズ時の再描画
        nv.utils.windowResize(function() { chart.update()});

        return chart;
});
}
```

課題－３

- step3.js を修正
  - 年月日の表記を変更
    - yyyy/mm/dd → yyyy-mm-dd
  - データの値を変えて、グラフの変化を確認
    - 日付に抜けがあると、どうなる？
  - データ件数を変えて、グラフの変化を確認

step4.js：時分秒データ、前半

```
window.onload = function() {
    // データのX軸を時分秒にする
    data1 = [{x:"00:00:00",y:1},{x:"00:01:00",y:2},{x:"00:02:00",y:3},{x:"00:03:00",y:4},{x:"00:04:00",y:5},];
    data2 = [{x:"00:00:00",y:20},{x:"00:01:00",y:19},{x:"00:02:00",y:18},{x:"00:03:00",y:17},{x:"00:04:00",y:1},];

    // 描画データを１つにまとめる
    var myData = [
        {       key: "data1", values: data1, color: "#ff0000",},
        {       key: "data2", values: data2, color: "#0000ff",},
    ];
    console.log(myData);            // 描画データをデバッグ出力
```

step4.js：年月日データ、後半

元データ：”00:00:00.000”
変換後　：”1970/01/01　00:00:00.000”
もっと複雑なデータ形式変換は、step6.js 参照

```
    nv.addGraph(function() {
        // グラフオブジェクト生成
        var chart = nv.models
                .lineChart()
                .x(function(d) { return Date.parse("1970/01/01 "+d.x) })// X軸データの取り出し方
                .y(function(d) { return d.y })                    //Y軸データの取り出し方
                .useInteractiveGuideline(true).showLegend(true).showYAxis(true).showXAxis(true);

        // x軸の表示形式設定（年月日表記）
        chart.xAxis.axisLabel('時分秒').tickFormat(function(d) {
                return d3.time.format('%H:%M:%S')(new Date(d))
        });
        // y軸の表示形式設定（数値表記）
        chart.yAxis.axisLabel('値').tickFormat(d3.format('d'));

        // グラフをHTMLの中に入れる
        d3.select("#view svg").datum(myData).call(chart);
        // リサイズ時の再描画
        nv.utils.windowResize(function() { chart.update()});

        return chart;
    });
}
```

課題－４

- step4.js を修正
  - X軸の変化の単位を変えてみる
    - １分ごと → １時間ごと
    - １分ごと → １秒ごと
  - データのX軸を変えて、グラフの変化を確認
    - data1：１分ごとのデータ
    - data2：30秒ごとのデータ
  - データ件数を変えて、グラフの変化を確認

step5.js：外部参照の練習、前半

```
window.onload = function() {
    // データを別リソースから読み込む
    var csvfile = "./step5.csv";
    d3.csv(csvfile,function(data) {
        // 描画データを１つにまとめる
        var myData = [
            {       key: "data1", values: data, color: "#ff0000",},
        ];
        console.log(myData);    // 描画データをデバッグ出力
```

step5.js：外部参照の練習、後半

```
    nv.addGraph(function() {
        // グラフオブジェクト生成
        var chart = nv.models
                .lineChart()
                .x(function(d) { return Date.parse("1970/01/01 "+d.x) })// X軸データの取り出し方
                .y(function(d) { return d.y })                    //Y軸データの取り出し方
                .useInteractiveGuideline(true).showLegend(true).showYAxis(true).showXAxis(true);

        // x軸の表示形式設定（年月日表記）
        chart.xAxis.axisLabel('時分秒').tickFormat(function(d) {
                return d3.time.format('%H:%M:%S')(new Date(d))
        });
        // y軸の表示形式設定（数値表記）
        chart.yAxis.axisLabel('値').tickFormat(d3.format('d'));

        // グラフをHTMLの中に入れる
        d3.select("#view svg").datum(myData).call(chart);
        // リサイズ時の再描画
        nv.utils.windowResize(function() { chart.update()});

        return chart;
    });
}
```

インデントが増えているだけで、step4.js とまったく同じ

step5.csv：外部参照の練習、データ

```
x,y
"00:00:00",1
"00:01:00",2
"00:02:00",3
"00:03:00",4
"00:04:00",5
```

課題ー5

- step5.csv を修正
  - X軸の変化の単位を変えてみる
    - １分ごと → １時間ごと
    - １分ごと → １秒ごと
  - データ件数を変えて、グラフの変化を確認

step6.js：外部参照、1

```
window.onload = function() {
    // もっと一般的な形のデータを読み込みたい
    var csvfile = "./step6.csv";
    d3.csv(csvfile,function(data) {
```

step6.js：外部参照、２

重要

```
// 描画用データに変換 ----------------------------------------------
var myData = new Array();
var maps = [     // データ変換テーブル
                {title:"data1", key:"y1", color:"#ff0000",},
                {title:"data2", key:"y2", color:"#0000ff",},
];
for (m in maps) {
                var values = new Array();
                for (d in data) {
                                var v = new Array();
                                v.x = data[d].x;
                                v.y = Number(data[d][maps[m].key]);          // 数値であることを明示
                                values.push(v);
                }
                var my = new Array();
                my.key = maps[m].title;
                my.values = values;
                my.color = maps[m].color;

                myData.push(my);
}
console.log(data);              // 元データをデバッグ出力
console.log(myData);            // 描画用データをデバッグ出力
// 描画用データに変換、ここまで ----------------------------------------------
```

step6.js：外部参照、３

```
    nv.addGraph(function() {
        // グラフオブジェクト生成
        var chart = nv.models
                .lineChart()
                .x(function(d) { return Date.parse("1970/01/01 "+d.x) })// X軸データの取り出し方
                .y(function(d) { return d.y })                    //Y軸データの取り出し方
                .useInteractiveGuideline(true).showLegend(true).showYAxis(true).showXAxis(true);

        // x軸の表示形式設定（年月日表記）
        chart.xAxis.axisLabel('時分秒').tickFormat(function(d) {
                return d3.time.format('%H:%M:%S')(new Date(d))
        });
        // y軸の表示形式設定（数値表記）
        chart.yAxis.axisLabel('値').tickFormat(d3.format('d'));

        // グラフをHTMLの中に入れる
        d3.select("#view svg").datum(myData).call(chart);
        // リサイズ時の再描画
        nv.utils.windowResize(function() { chart.update()});

        return chart;
    });
}
```

step5.js の後半とまったく同じ

step6.csv：外部参照、データ

```
x,y1,y2
"00:00:00",1,20
"00:01:00",2,19
"00:02:00",3,18
"00:03:00",4,17
"00:04:00",5,1
```

課題ー6

- step6.js を修正
  - 気象庁サイトから気温のCSVを入手し、可視化
    - http://www.data.jma.go.jp/gmd/risk/obsdl/
    - tokyo2015.csv
      - 地点：東京
      - 項目：日平均気温、日最高気温、日最低気温
      - 期間：2015/01/01 ～ 2016/01/01
      - ダウンロード後に、ヘッダを書き変え済み
  - Internet上のAPIでデータを取得し、可視化
    - http://aramoto.sakura.ne.jp/IoT/details.php?userId=&deviceId=&dataType=TEMP&startDate=&endDate=&scale=MINUTE&limit=1000
    - scaleをMINUTE→HOURやDAYに変更してみる

step7.js：クリック位置取得

```
    nv.addGraph(function() {
        // グラフオブジェクト生成
        var chart = nv.models
                .lineChart()
                .x(function(d) { return Date.parse("1970/01/01 "+d.x) })// X軸データの取り出し方
                .y(function(d) { return d.y })                    //Y軸データの取り出し方
                .useInteractiveGuideline(true).showLegend(true).showYAxis(true).showXAxis(true);

        // x軸の表示形式設定（年月日表記）
        <<省略>>
        // グラフをHTMLの中に入れる
        d3.select("#view svg").datum(myData).call(chart);
        // リサイズ時の再描画
        nv.utils.windowResize(function() { chart.update()});

        // クリックした位置を知りたい
        chart.lines.dispatch.on("elementClick", function(e) {
                console.log(e);
        });

        return chart;
    });
}
```

後編
センサでデータを取って可視化してみよう

センサからのデータを可視化

- 必要なもの
  - PC + Arduino
  - Raspberry PI
  - 各種センサ
- 各種センサを購入できるサイト
  - スイッチサイエンス
    - https://www.switch-science.com/
  - 秋月電子通商
    - http://akizukidenshi.com/catalog/default.aspx
  - Amazon
    - https://www.amazon.co.jp/

Arduinoの使い方

- 開発環境をダウンロード
  - https://www.arduino.cc/en/Main/Software
- Windowsはドライバをインストール
  - 開発環境の「drivers/arduino.info」を右クリック
  - デバイスマネージャでCOMを確認
- Macは何もしなくても良い

- 詳しく知りたい人は、こちら
  - http://cloud.aitc.jp/20140624_Arduino/
  - http://cloud.aitc.jp/20150801_Arduino/

Arduino：アナログ0を読んで出力

```
void setup() {
    Serial.begin(9600);
}

long count = 0;
void loop() {
    count++;
    Serial.print (count);
    Serial.print (“,”);
    int value = analogRead(0);
    Serial.println(value);

    delay(1000);
}
```

Arduinoは時計を持っていないので、
日時は出力できない

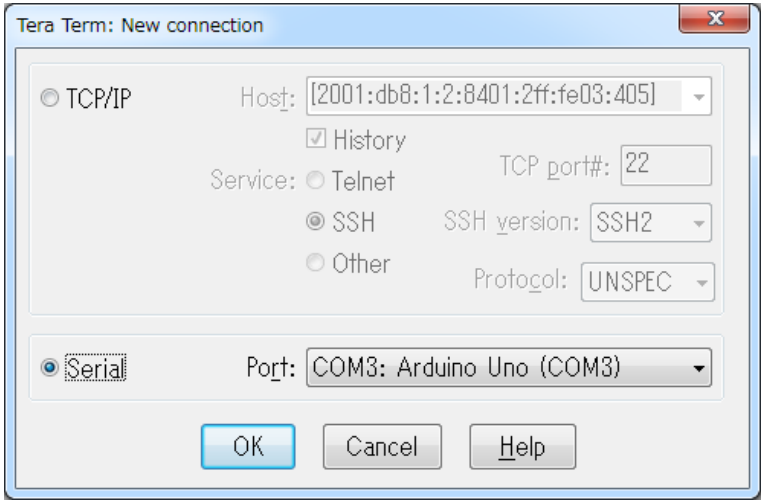
step6.js：修正点

```
    nv.addGraph(function() {
        // グラフオブジェクト生成
        var chart = nv.models
                .lineChart()
                .x(function(d) { return Number(d.x) })      // X軸データの取り出し方
                .y(function(d) { return d.y })              //Y軸データの取り出し方
                .useInteractiveGuideline(true).showLegend(true).showYAxis(true).showXAxis(true);

        // x軸の表示形式設定（年月日表記）
        chart.xAxis.axisLabel('時分秒').tickFormat(function(d) {
                return d3.time.format('%H:%M:%S')(new Date(d))
        });
        // y軸の表示形式設定（数値表記）
        chart.yAxis.axisLabel('値').tickFormat(d3.format('d'));

        // グラフをHTMLの中に入れる
        d3.select("#view svg").datum(myData).call(chart);
        // リサイズ時の再描画
        nv.utils.windowResize(function() { chart.update()});

        return chart;
    });
}
```

削除

センサ値取得：Windows編

- シリアルポートに接続可能なソフトを導入
  - 例：TeraTerm
    - シリアルポートに接続する
    - メニューの「File」→「Log...」を指定
      - 可視化対象のファイル名を指定
    - いったん止めて、先頭にCSVヘッダを追記
    - ログの収集を再開



センサ値取得：Mac編

- ポートが「/dev/cu.usbmodemfa131」の場合
  - ターミナルを開き、以下のコマンドを実行
    - echo “x,y1” > aaa.csv
    - cat　/dev/cu.usbmodemfa131 >> aaa.csv

コピペは危険

今後：本格運用するためには

- WebAPIを作成
  - データ受信
    - 受信したデータをDBに格納する
  - データ参照
    - 指定した範囲のデータをDBから取り出す

- WebAPIを作成するために
  - Webサーバで使える開発言語を勉強
    - PHP, Perl, python, Java, Ruby, e.t.c....

可視化してみよう
～17:45

色々なセンサを
使ってみましょう。